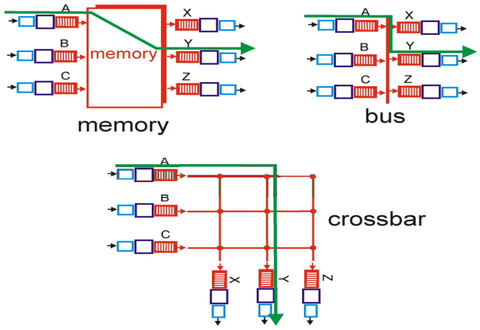
CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 10 -
Intra-Domain Routing

From previous lecture….
Three Types of Switching Fabrics

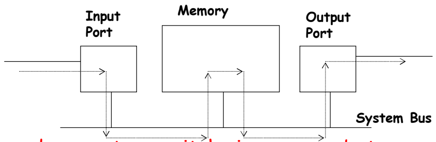


2

Switching Via a Memory

First generation routers → looked like PCs

- Packet copied by system's (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)



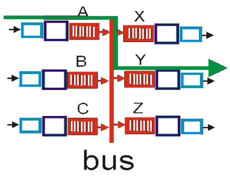
Most modern routers switch via memory, but…

- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

3

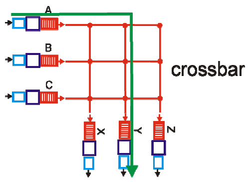
Switching Via a Bus

- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



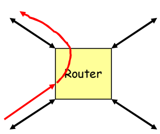
Switching Via an Interconnection Network

- Overcome bus and memory bandwidth limitations
- Crossbar provides full NxN interconnect
  - Expensive
  - Uses 2N buses
- Cisco 12000: switches Gbps through the interconnection network



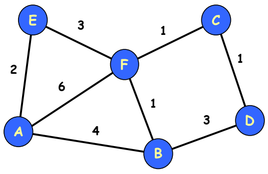
Routing

- Past two lectures
  - IP addresses are structured
  - IP packet headers carry these addresses
  - When packet arrives at router
    - Examine header for intended destination
    - Look up next hop in table
    - Send packet out appropriate port
- This lecture:
  - How these forwarding tables are built?
  - Routing algorithms



A Model of the Problem

- Network as a Graph:
  - Represent each router as node
  - Direct link between routers represented by edge
  - Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of difficulty of using link
  - delay, $ cost, or congestion level
- Task
  - Determine *least cost path* from every node to every other node
    - Path cost d(x,y) = sum of link costs



7

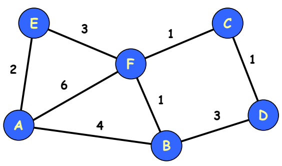
Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Distributed
  - Routers perform local computation
  - Converge to a globally consistent routing state
  - *"Global": Link-state*
    - Every node collects complete graph structure
    - Each computes shortest paths from it
    - Each generates own routing table
  - *Local: Distance-vector*
    - No one has copy of graph
    - Nodes construct their own tables iteratively
    - Each sends information about its table to neighbors

8

Distance-Vector Method

Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F



- Idea
  - At any time, have cost/next hop of best known path to destination
  - Use cost ∞ when no path known
- Initially
  - Only have entries for directly connected nodes

9

Algorithm

Each node x stores:
- c(x,v) for each neighbor v
- Distance vector of node x: estimate of d(x,y) for all y
- Distance vectors heard from each neighbor

Initialization:
1. d(x,y) = c(x,y) for all y.
2. Send distance vector to each neighbor

Repeat:

Whenever link cost to neighbor changes or distance vector received from neighbor

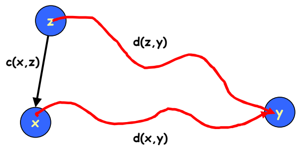
For every neighbor z

For every destination y

d(x,y) ← Update(x,y,z)

If d(x,y) changed for any y, send distance vector to all neighbors

Distance-Vector Update



Update(x,y,z)

d ← c(x,z) + d(z,y)   /* Cost of path from x to y with first hop z */

if d < d(x,y)

/* Found better path */

return d,z   /* Updated cost / next hop */

else

return d(x,y), nexthop(x,y)   /* Existing cost / next hop */

Start

Optimum 1-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	∞	-	C	∞	-
D	∞	-	D	3	D
E	2	E	E	∞	-
F	6	F	F	1	F

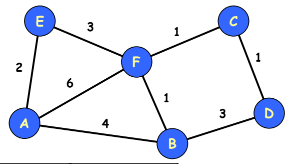


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F

Iteration #1

Optimum 2-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

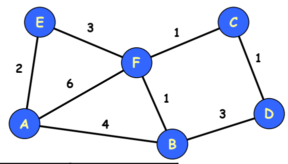
Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

Table for C		
Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	-
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	-
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F



13

Iteration #2

Optimum 3-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

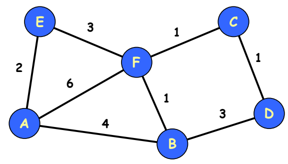
Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

Table for C		
Dst	Cst	Hop
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

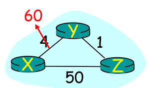


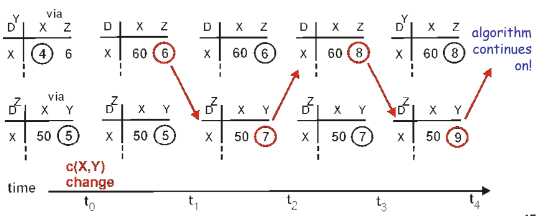
14

Distance Vector: Link Cost Changes

Link cost changes:

- Bad news travels slow - "count to infinity" problem!

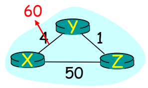


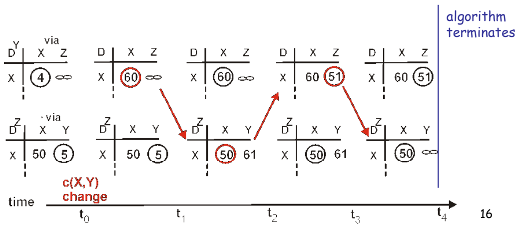


15

Distance Vector: Poison Reverse

If Z routes through Y to get to X :
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Will this completely solve count to infinity problem?





16

Poison Reverse Failures

Table for A			Table for B			Table for D			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
C	7	F	C	8	A	C	9	B	C	1	C

Forced Update

Table for A		
Dst	Cst	Hop
C	∞	-

Forced Update

Table for F		
Dst	Cst	Hop
C	∞	-

Better Route

Table for A		
Dst	Cst	Hop
C	13	D

Forced Update

Table for B		
Dst	Cst	Hop
C	14	A

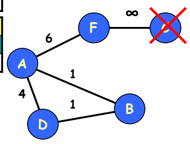
Forced Update

Table for D		
Dst	Cst	Hop
C	15	B

Forced Update

Table for A		
Dst	Cst	Hop
C	19	D

.
.
.



- Iterations don't converge
- "Count to infinity"
- Solution
  - Make "infinity" smaller
  - What is upper bound on maximum path length?

17

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
  - Current standard is version 2 (RFC 1723)
- Features
  - Every link has cost 1 → Hop count
  - "Infinity" = 16
    - Limits to networks where everything reachable within 15 hops
- Sending Updates
  - Every router listens for updates on UDP port 520
  - Triggered
    - When every entry changes, send copy of entry to neighbors
      - Except for one causing update (split horizon rule)
  - Periodic
    - Every 30 seconds, router sends copy of its table to each neighbor

18

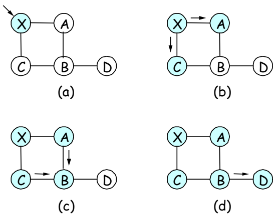
Link State Protocol Concept

- Every node gets complete copy of graph
  - Every node "floods" network with data about its outgoing links
- Every node computes routes to every other node
  - Using single-source, shortest-path algorithm
- Process performed whenever needed
  - When interconnections die / reappear

19

Sending Link States by "Flooding"

- X wants to send information
  - Sends on all outgoing links
- When node Y receives information from Z
  - Resend on all links other than Z



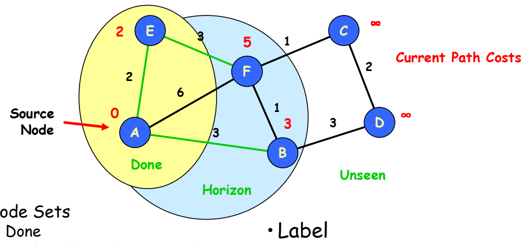
20

Dijkstra's Algorithm

- Given
  - Graph with source node s and edge costs c(u,v)
  - Determine least cost path from s to every node v
- Single source shortest Path Algorithm
  - Traverse graph in order of least cost from source

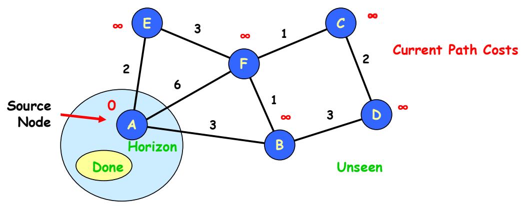
21

Dijkstra's Algorithm



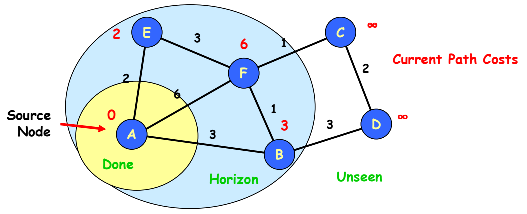
- Node Sets
  - Done
    - Already have least cost path to it
  - Horizon:
    - Reachable in 1 hop from node in Done
  - Unseen:
    - Cannot reach directly from node in Done
- Label
  - d(v) = path cost
    - From s to v
- Path
  - Keep track of last link in path

Dijkstra's Algorithm: Initially



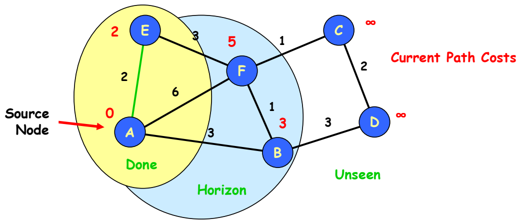
- No nodes "done"
- Source in "horizon"

Dijkstra's Algorithm: Initially



- d(v) to node A shown in red
  - Only consider links from done nodes

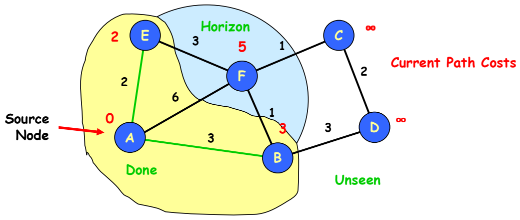
Dijkstra's Algorithm



- Select node v in horizon with minimum d(v)
- Add link used to add node to shortest path tree
- Update d(v) information

25

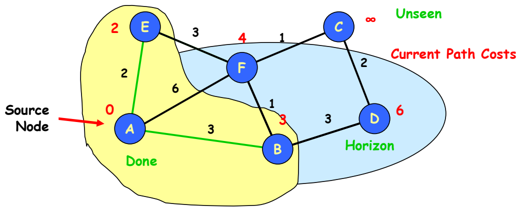
Dijkstra's Algorithm



- Repeat…

26

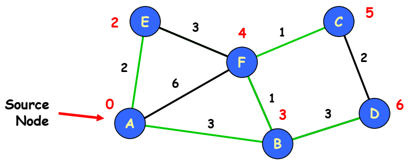
Dijkstra's Algorithm



- Addition of node can add new nodes to horizon

27

Dijkstra's Algorithm

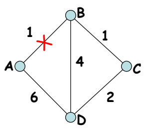


- Final tree shown in green

28

Link State Characteristics

- With consistent LSDBs*, all nodes compute consistent loop-free paths
- Can still have transient loops



Packet from C→A may loop around BDC if B knows about failure and C & D do not

*Link State Data Base

29

OSPF Routing Protocol

- Open
  - Open standard created by IETF
- More prevalent than RIP

30

OSPF Messages

- Transmit link state advertisements
  - Originating router
    - Typically, IP address for router
  - Link ID
    - ID of router at other end of link
  - Metric
    - Cost of link
  - Sequence number
    - Incremented each time sending new link information

31

OSPF Flooding Operation

- Node X Receives LSA from Node Y
  - With Sequence Number q
  - Looks for entry with same origin/link ID
- Cases
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number $p < q$
    - Update entry, propagate to all neighbors other than Y
  - Entry present with sequence number $p > q$
    - Send entry back to Y
    - To tell Y that it has out-of-date information
  - Entry present with sequence number $p = q$
    - Ignore it

32

Flooding Issues

- When should it be performed
  - Periodically
  - When status of link changes
    - Detected by connected node
    - Congestion, lack of electric or optical signal
- What happens when router goes down & back up
  - Sequence number reset to 0
    - Other routers may have entries with higher sequence numbers
  - Router will send out LSAs with number 0
  - Will get back LSAs with last valid sequence number p
  - Router sets sequence number to p+1 & resends

33

Adoption of OSPF

- RIP viewed as outmoded
  - Good when networks small and routers had limited memory & computational power
- OSPF Advantages
  - Fast convergence when configuration changes
  - Full topology map helps

34

Comparison of LS and DV Algorithms

Message complexity

- LS: with n nodes, v neighbors, O(nv) messages per node
- DV: exchange between neighbors only

Speed of Convergence

- LS: Complex computation
  - But…can forward before computation
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem
  - (faster with triggered updates)

35

Comparison of LS and DV Algorithms

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - errors propagate thru network
- Other tradeoffs
  - Making LSP flood reliable difficult
  - Prioritize routing packets?

36